

Learning Onto-Relational Rules with Inductive Logic Programming

Francesca A. LISI^{a,1},

^a *Dipartimento di Informatica, Università degli Studi di Bari "Aldo Moro", Italy*

Abstract. Rules complement and extend ontologies on the Semantic Web. We refer to these rules as onto-relational since they combine DL-based ontology languages and Knowledge Representation formalisms supporting the relational data model within the tradition of Logic Programming and Deductive Databases. Rule authoring is a very demanding Knowledge Engineering task which can be automated though partially by applying Machine Learning algorithms. In this chapter we show how Inductive Logic Programming (ILP), born at the intersection of Machine Learning and Logic Programming and considered as a major approach to Relational Learning, can be adapted to Onto-Relational Learning. For the sake of illustration, we provide details of a specific Onto-Relational Learning solution to the problem of learning rule-based definitions of DL concepts and roles with ILP.

Keywords. Inductive Logic Programming, Rule Languages and Systems, Integration of Rules and Ontologies, Deductive Databases.

Introduction

Rules are widely used in Knowledge Engineering (KE) and Knowledge Representation (KR) as a powerful way of modeling knowledge. In the broadest sense, a rule could be any statement which says that a certain conclusion must be valid whenever a certain premise is satisfied, *i.e.* any statement that could be read as a sentence of the form “if .. then ..”. Rules have been successfully applied in the fields of Logic Programming (LP) and Deductive Databases [6]. Rules play also a role in the *Semantic Web* architecture. Interest in this area has grown rapidly over recent years as testified by the Rules Interchange Format (RIF)² activity at W3C. Rules from the RIF perspective would allow the integration, transformation and derivation of data from numerous sources in a distributed, scalable, and transparent manner. Because of the great variety in rule languages and rule engine technologies, RIF consists of a core language³ to be used along with a set of standard and non-standard extensions. These extensions need not all be combinable into a single unified language. As for the expressive power, two directions are followed: monotonic extensions towards full First Order Logic (FOL) and non-monotonic (NM) extensions based on the LP tradition. The debate around a RIF has taken a long time also

¹Corresponding Author: Francesca A. Lisi, Dipartimento di Informatica, Università degli Studi di Bari “Aldo Moro”, Italy; E-mail: lisi@di.uniba.it

²http://www.w3.org/2005/rules/wiki/RIF_Working_Group

³<http://www.w3.org/TR/rif-core/>

due to the controversial issue of having rules on top or aside ontologies [19]. There is a consensus now on the fact that rules complement and extend ontologies. Indeed, rules can be used in combination with ontologies, or as a means to specify ontologies. They are also frequently applied over ontologies, to draw inferences, express constraints, specify policies, react to events, discover new knowledge, transform data, etc. In particular, RIF rules can refer to RDF and OWL facts. Since the design of OWL has been based on the \mathcal{SH} family of very expressive *Description Logics* (DLs) (see Chapter ?? for an introduction), the NM dialects of RIF will most likely be inspired by those hybrid KR systems that integrate DLs and LP. Such rule formalisms are of interest to this chapter. We shall refer to them as *onto-relational rule languages* from now on. Apart from the specific ontology language, the integration of ontologies and rules is already present in existing knowledge bases (KBs). Notably the Cyc⁴ KB consists of terms (which constitute the vocabulary, *i.e.* the ontology) and assertions which relate those terms and include both simple ground assertions and rules [22].

The acquisition of rules for very large KBs like Cyc is a very demanding KE activity. Indeed, according to an estimate from the Cyc project, human experts produce rules at the rate of approximately three per hour but can evaluate an average of twenty rules per hour. Also, for untrained knowledge engineers, while rule authoring may be very difficult, rule reviewing is feasible (although still difficult). A partial automation of the rule authoring task, *e.g.* by applying *Machine Learning* (ML) algorithms (see Chapter ?? for an introduction), can be of help even though the automatically produced rules are not guaranteed to be correct. In fact, of those rules, some will turn out to be correct, and some will be found to need editing to be assertible. Yet, as mentioned above, rule reviewing is less critical than rule authoring. In order to partially automate the authoring of onto-relational rules, the bunch of ML techniques collectively known under the name of *Inductive Logic Programming* (ILP) [40] seems particularly promising for the following reasons. ILP was born at the intersection of ML and LP [39], and is widely recognized as a major approach to *Relational Learning* [7]. Apart from the KR framework of LP, the distinguishing feature of ILP, also with respect to other ML forms, is the use of prior domain knowledge in the form of a logical theory during the induction process. In this chapter we take a critical look at ILP proposals for learning relational rules while having an ontology as the background theory. These proposals try to overcome the difficulties of accommodating ontologies in Relational Learning. The work of [3] on using semantic meta-knowledge from Cyc as inductive bias in an ILP system is another attempt at solving this problem though more empirically. Conversely, we promote an extension of Relational Learning, called *Onto-Relational Learning* (ORL), which accounts for ontologies in a clear, elegant and well-founded manner by resorting to onto-relational rule languages. In this chapter, for the sake of illustration, we provide details of a specific ORL solution to the problem of learning rule-based definitions of DL concepts and roles with ILP.

The chapter is organized as follows. Section 1 is devoted to preliminaries on LP and its applications to databases and ontologies as well as on ILP. Section 2 provides a state-of-the-art survey of ILP proposals for learning onto-relational rules. Section 3 describes in depth the most powerful of these proposals. Section 4 concludes the chapter with final remarks and outlines directions of future work.

⁴http://cyc.com/cyc/technology/whatis_cyc_dir/

1. Preliminaries

1.1. Logic Programming and databases

Logic Programming (LP) is rooted into a fragment of Clausal Logics (CLs) known as Horn Clausal Logic (HCL) [34]. The basic element in CLs is the *atom* of the form $p(t_i, \dots, t_{k_i})$ such that each p is a predicate symbol and each t_j is a term. A *term* is either a constant or a variable or a more complex term obtained by applying a functor to simpler term. Constant, variable, functor and predicate symbols belong to mutually disjoint alphabets. A *literal* is an atom either negated or not. A *clause* is a universally quantified disjunction of literals. Usually the universal quantifiers are omitted to simplify notation. Alternative notations are a clause as set of literals and a clause as an implication. A *program* is a set of clauses. HCL admits only so-called definite clauses. A *definite clause* is an implication of the form

$$\alpha_0 \leftarrow \alpha_1, \dots, \alpha_m$$

where $m \geq 0$ and α_i are atoms, *i.e.* a clause with exactly one positive literal. The right-hand side α_0 and the left-hand side $\alpha_1, \dots, \alpha_m$ of the implication are called *head* and *body* of the clause, respectively. Note that the body is intended to be an existentially quantified conjunctive formula $\exists \alpha_1 \wedge \dots \wedge \alpha_m$. Furthermore definite clauses with $m > 0$ and $m = 0$ are called *rules* and *facts* respectively. The model-theoretic semantics of HCL is based on the notion of *Herbrand interpretation*, *i.e.* an interpretation in which all constants and function symbols are assigned very simple meanings. This allows the symbols in a set of clauses to be interpreted in a purely syntactic way, separated from any real instantiation. The corresponding proof-theoretic semantics is based on the *Closed World Assumption* (CWA), *i.e.* the presumption that what is not currently known to be true, is false. Deductive reasoning with HCL is formalized in its proof theory. In clausal logic *resolution* comprises a single inference rule which, from any two clauses having an appropriate form, derives a new clause as their consequence. Resolution is sound: every resolvent is implied by its parents. It is also refutation complete: the empty clause is derivable by resolution from any set S of Horn clauses if S is unsatisfiable. *Negation As Failure* (NAF) is related to the CWA, as it amounts to believing false every predicate that cannot be proved to be true. Clauses with NAF literals in the body are called *normal clauses*. The concept of a *stable model*, or *answer set*, is used to define a declarative semantics for normal logic programs [16]. According to this semantics, a logic program may have several alternative models (but possibly none), each corresponding to a possible view of the reality. Also based on the stable model (answer set) semantics, *Answer Set Programming* (ASP) is an alternative LP paradigm oriented towards difficult search problems [35].

Definite clauses played a prominent role in the rise of deductive databases [6]. More precisely, functor-free non-recursive definite clauses are at the basis of the language Datalog for deductive databases [5]. Generally, it is denoted by DATALOG^\neg where \neg is treated as NAF. The restriction of Datalog to only positive rules (*i.e.*, rules without NAF literals) is denoted by DATALOG . Based on the distinction between extensional and intensional predicates, a DATALOG program Π can be divided into two parts. The *extensional part*, denoted as $\text{EDB}(\Pi)$, is the set of facts of Π involving the extensional predicates, whereas the *intensional part* $\text{IDB}(\Pi)$ is the set of all other clauses of Π . The main

reasoning task in DATALOG is *query answering*. A *query* Q to a DATALOG program Π is a DATALOG clause of the form

$$\leftarrow \alpha_1, \dots, \alpha_m$$

where $m > 0$, and α_i is a DATALOG atom. An *answer* to a query Q is a substitution θ for the variables of Q . An answer is correct with respect to the DATALOG program Π if $\Pi \models Q\theta$. The *answer set* to a query Q is the set of answers to Q that are correct w.r.t. Π and such that $Q\theta$ is ground. In other words the answer set to a query Q is the set of all ground instances of Q which are logical consequences of Π . Answers are computed by refutation.

Disjunctive Datalog (denoted as DATALOG^\vee) is a variant of DATALOG where disjunctions may appear in the rule heads [10]. Therefore DATALOG^\vee can not be considered as a fragment of HCL. Advanced versions ($\text{DATALOG}^{\neg\vee}$) also allow for negation in the bodies, which can be handled according to a semantics for negation in CLs. Defining the semantics of a $\text{DATALOG}^{\neg\vee}$ program is complicated by the presence of disjunction in the rules' heads because it makes the underlying disjunctive logic programming inherently nonmonotonic, *i.e.* new information can invalidate previous conclusions. Among the many alternatives, one widely accepted semantics for $\text{DATALOG}^{\neg\vee}$ is the extension of the stable model semantics to the disjunctive case.

1.2. Logic Programming and ontologies

The integration of LP and ontologies follows the tradition of KR research on so-called *hybrid systems*, *i.e.* those systems which are constituted by two or more subsystems dealing with distinct portions of a single KB by performing specific reasoning procedures [15]. The motivation for investigating and developing such systems is to improve on two basic features of KR formalisms, namely *representational adequacy* and *deductive power*, by preserving the other crucial feature, *i.e. decidability*. Indeed DLs and CLs are FOL fragments incomparable as for the expressiveness [1] and the semantics [43] but combinable at different degrees of integration: Tight, loose, full.

The semantic integration is *tight* when a model of the hybrid KB is defined as the union of two models, one for the DL part and one for the CL part, which share the same domain. In particular, combining DLs with CLs in a tight manner can easily lead to undecidability if the interaction scheme between the DL and the CL part of a hybrid KB does not solve the semantic mismatch between DLs and CLs [44]. This requirement is known as *DL-safety* [38]. With respect to this property, the hybrid KR system CARIN [23] is *unsafe* because the interaction scheme is left unrestricted. Conversely, \mathcal{AL} -LOG [8] guarantees a *safe* interaction scheme by means of syntactic restrictions. Finally, $\mathcal{DL}+\text{LOG}^{\neg\vee}$ [45]⁵ is *weakly DL-safe* because it relaxes the condition of DL-safety. The distinguishing features of these three KR frameworks are summarized in Table 1 and further discussed in Section 1.2.1, 1.2.2, and 1.2.3 respectively.

The semantic integration is *loose* when the DL part and the CL part are separate components connected through a minimal interface for exchanging knowledge. An example of one such kind of coupling is the integration scheme for ASP and DLs illustrated

⁵We prefer $\mathcal{DL}+\text{LOG}^{\neg\vee}$ to the original name $\mathcal{DL}+\text{LOG}$ in order to emphasize the NM features of the language.

Table 1. Three KR frameworks suitable for representing onto-relational rules.

	CARIN [23]	$\mathcal{AL}\text{-LOG}$ [8]	$\mathcal{DL}\text{-LOG}^{\neg\vee}$ [45]
DL language	any DL	\mathcal{ALC}	any DL
CL language	Horn clauses	DATALOG clauses	DATALOG $^{\neg\vee}$ clauses
integration	tight DL-unsafe	tight DL-safe	tight weakly DL-safe
rule head literals	DL/Horn literals	DATALOG literal	DL/DATALOG literals
rule body literals	DL/Horn literals	\mathcal{ALC} /DATALOG literals (no roles)	DL/DATALOG $^{\neg}$ literals
semantics	Herbrand models+DL models	idem	stable models+DL models
reasoning	SLD-resolution+tableau calculus	idem	stable model computation + Boolean CQ/UCQ containment
decidability	only for some instantiations	yes	for all instantiations with DLs for which the Boolean CQ/UCQ containment is decidable
implementation	yes, e.g.[18]	yes, e.g.[48]	unknown

in [11]. It derives from the previous work of the same authors on the extension of ASP with higher-order reasoning and external evaluations [12] which has been implemented into the system DLVHEX⁶.

The semantic integration is *full* when there is no separation between vocabularies of the two parts of the hybrid KB. One such kind of coupling is achieved by means of the logic of Minimal Knowledge and Negation as Failure in [37].

A complete picture of the computational properties of systems combining DL ontologies and DATALOG rules can be found in [46]. An updated survey of the literature on hybrid DL-CL systems [9] is suggested for further reading.

1.2.1. CARIN

A comprehensive study of the effects of combining DLs and CLs (more precisely, Horn rules) can be found in [23]. Special attention is devoted to the DL $\mathcal{ALCN}\mathcal{R}$. The results of the study can be summarized as follows: (i) answering conjunctive queries over $\mathcal{ALCN}\mathcal{R}$ TBoxes is decidable, (ii) query answering in $\mathcal{ALCN}\mathcal{R}$ extended with non-recursive DATALOG rules, where both concepts and roles can occur in rule bodies, is also decidable, as it can be reduced to answering a *union of conjunctive queries* (UCQ)⁷, (iii) if rules are recursive, query answering becomes undecidable, (iv) decidability can be regained by disallowing certain combinations of constructors in the logic, and (v) decidability can be regained by requiring rules to be *role-safe*, where at least one variable from each role literal must occur in some non-DL-atom. The integration framework proposed in [23] and known as CARIN is therefore DL-unsafe. Reasoning in CARIN is based on *constrained SLD-resolution*, i.e. an extension of SLD-resolution with a tableau calculus for DLs to deal with DL literals in the rules. Constrained SLD-refutation is a complete and sound method for answering *ground* queries.

⁶<http://www.kr.tuwien.ac.at/research/systems/dlvhex/>

⁷A UCQ over a predicate alphabet P is a FOL sentence of the form $\exists \vec{X}. conj_1(\vec{X}) \vee \dots \vee conj_n(\vec{X})$, where \vec{X} is a tuple of variable symbols and each $conj_i(\vec{X})$ is a set of atoms whose predicates are in P and whose arguments are either constants or variables from \vec{X} . A CQ is a UCQ with $n = 1$.

1.2.2. \mathcal{AL} -LOG

\mathcal{AL} -LOG is a hybrid KR system that integrates safely the DL \mathcal{ALC} and DATALOG [8]. In particular, variables occurring in the body of rules may be constrained with \mathcal{ALC} concept assertions to be used as 'typing constraints'. This makes rules applicable only to explicitly named objects. As in CARIN, query answering is decided using the constrained SLD-resolution which however in \mathcal{AL} -LOG is decidable and runs in single non-deterministic exponential time.

1.2.3. $\mathcal{DL}+\text{LOG}^{\neg\vee}$

The hybrid KR framework of $\mathcal{DL}+\text{LOG}^{\neg\vee}$ allows a \mathcal{DL} KB, *i.e.* a KB expressed in any DL, to be extended with weakly DL-safe $\text{DATALOG}^{\neg\vee}$ rules [45]. Weak DL-safeness allows to overcome the main representational limits of the DL-safe approaches, *e.g.* the possibility of expressing UCQs, by keeping the integration scheme still decidable. For $\mathcal{DL}+\text{LOG}^{\neg\vee}$ two semantics have been defined: a FOL semantics and a NM semantics. In particular, the latter extends the stable model semantics of $\text{DATALOG}^{\neg\vee}$. According to it, \mathcal{DL} -predicates are still interpreted under OWA, while DATALOG-predicates are interpreted under CWA. Notice that, under both semantics, entailment can be reduced to satisfiability and, analogously, that CQ answering can be reduced to satisfiability. The problem statement of satisfiability for finite $\mathcal{DL}+\text{LOG}^{\neg\vee}$ KBs relies on the problem known as the *Boolean CQ/UCQ containment problem*⁸ in \mathcal{DL} . It is shown that the decidability of reasoning in $\mathcal{DL}+\text{LOG}^{\neg\vee}$, thus of ground query answering, depends on the decidability of the Boolean CQ/UCQ containment problem in \mathcal{DL} . Currently, *SHIQ* is one of the most expressive DLs for which this problem is decidable [17].

1.3. Inductive Logic Programming

Inductive Logic Programming (ILP) was born at the intersection between LP and ML [39]. From LP it has borrowed the KR framework, *i.e.* HCL. From ML (more precisely, from Concept Learning) it has inherited the inferential mechanisms for induction, the most prominent of which is *generalization*. However, a distinguishing feature of ILP with respect to other forms of Concept Learning is the use of prior knowledge of the domain of interest, called *background knowledge* (BK). Therefore, induction with ILP generalizes from individual instances/observations in the presence of BK, finding valid hypotheses. *Validity* depends on the underlying *setting*. At present, there exist several formalizations of induction in ILP that can be classified according to the following two orthogonal dimensions: the *scope of induction* (discrimination vs characterization) and the *representation of observations* (ground definite clauses vs ground unit clauses). *Discriminant induction* aims at inducing hypotheses with discriminant power as required in tasks like classification. In classification, observations encompass both positive and negative examples. *Characteristic induction* is more suitable for finding regularities in a data set. This corresponds to learning from positive examples only. The second dimension affects the notion of *coverage*, *i.e.* the condition under which a hypothesis explains an observation. In *learning from entailment*, hypotheses are clausal theories, observations are ground definite clauses, and a hypothesis covers an observation if the hypothesis logically entails the observation. In *learning from interpretations*, hypotheses are clausal the-

⁸This problem was called *existential entailment* in [23].

ories, observations are Herbrand interpretations (ground unit clauses) and a hypothesis covers an observation if the observation is a model for the hypothesis.

In Concept Learning, generalization is traditionally viewed as search through a partially ordered space of inductive hypotheses [36]. According to this vision, an inductive hypothesis in ILP is a clausal theory and the induction of a single clause requires (i) structuring, (ii) searching and (iii) bounding the space of clauses [40]. First we focus on (i) by clarifying the notion of *ordering* for clauses. An ordering allows for determining which one, between two clauses, is more general than the other. Since partial orders are considered, uncomparable pairs of clauses are admitted. Given the usefulness of BK, orders have been proposed that reckon with it. Among them, *generalized subsumption* [2] is of major interest to this chapter: Given two definite clauses C and D standardized apart and a definite program \mathcal{K} , we say that $C \succeq_{\mathcal{K}} D$ iff there exists a ground substitution θ for C such that (i) $head(C)\theta = head(D)\sigma$ and (ii) $\mathcal{K} \cup body(D)\sigma \models body(C)\theta$ where σ is a Skolem substitution for D with respect to $\{C\} \cup \mathcal{K}$. Generalized subsumption is also called *semantic generality* in contrast to other orders which are purely syntactic. In the general case, it is undecidable. However, for DATALOG it is decidable and admits a least general generalization. Once structured, the space of hypotheses can be searched (ii) by means of refinement operators. A *refinement operator* is a function which computes a set of specializations or generalizations of a clause according to whether a top-down or a bottom-up search is performed. The two kinds of refinement operator have been therefore called *downward* and *upward*, respectively. The definition of refinement operators presupposes the investigation of the properties of the various orderings and is usually coupled with the specification of a declarative bias for bounding the space of clauses (iii). *Bias* concerns anything which constrains the search for theories, e.g. a *language bias* specifies syntactic constraints such as *linkedness* and *connectedness* on the clauses in the search space. A definite clause C is linked if each literal $l_i \in C$ is linked. A literal $l_i \in C$ is linked if at least one of its terms is linked. A term t in some literal $l_i \in C$ is linked with linking-chain of length 0, if t occurs in $head(C)$, and with linking-chain of length $d + 1$, if some other term in l_i is linked with linking-chain of length d . The link-depth of a term t in l_i is the length of the shortest linking-chain of t . A clause C is connected if each variable occurring in $head(C)$ also occurs in $body(C)$.

2. ILP for Onto-Relational Rule Learning: State of the Art

Hybrid KR systems combining DLs and CLs with a tight integration scheme have very recently attracted some attention in the ILP community: [47] chooses CARIN- \mathcal{ALN} , [24] resorts to \mathcal{AL} -LOG, and [27] builds upon $SHIQ$ +LOG. A comparative analysis of the three is reported in Table 2. They can be considered as attempts at accommodating ontologies in ILP. Indeed, they can deal with \mathcal{ALN} , \mathcal{ALC} , and $SHIQ$ ontologies respectively. We remind the reader that \mathcal{ALN} and \mathcal{ALC} are incomparable DLs whereas DLs in the SH family enrich \mathcal{ALC} with further constructors.

Closely related to KR systems integrating DLs and CLs are the hybrid formalisms arising from the study of many-sorted logics, where a FOL language is combined with a sort language which can be regarded as an elementary DL [13]. In this respect the study of a sorted downward refinement [14] can be also considered as a contribution to the problem of interest to this chapter. Finally, some work has been done on discovering frequent association patterns in the form of DL-safe rules [20].

2.1. Learning CARIN- \mathcal{ALN} rules

The framework proposed in [47] focuses on discriminant induction and adopts the ILP setting of learning from interpretations. Hypotheses are represented as CARIN- \mathcal{ALN} non-recursive rules with a Horn literal in the head that plays the role of target concept. The coverage relation of hypotheses against examples adapts the usual one in learning from interpretations to the case of hybrid CARIN- \mathcal{ALN} BK. The generality relation between two hypotheses is defined as an extension of generalized subsumption. Procedures for testing both the coverage relation and the generality relation are based on the existential entailment algorithm of CARIN. Following [47], Kietz studies the learnability of CARIN- \mathcal{ALN} , thus providing a pre-processing method which enables ILP systems to learn CARIN- \mathcal{ALN} rules [21].

2.2. Learning \mathcal{AL} -LOG rules

In [24], hypotheses are represented as constrained DATALOG clauses that are linked, connected (or range-restricted), and compliant with the bias of Object Identity (OI)⁹. Unlike [47], this framework is general, meaning that it is valid whatever the scope of induction is. The generality relation for one such hypothesis language is an adaptation of generalized subsumption, named \mathcal{B} -subsumption, to the \mathcal{AL} -LOG KR framework. It gives rise to a quasi-order and can be checked with a decidable procedure based on constrained SLD-resolution [30]. Coverage relations for both ILP settings of learning from interpretations and learning from entailment have been defined on the basis of query answering in \mathcal{AL} -LOG [26]. As opposed to [47], the framework has been implemented in an ILP system [32,33]. More precisely, an instantiation of it for the case of *characteristic induction from interpretations* has been considered. Indeed, the system supports a variant of a very popular data mining task - frequent pattern discovery - where rich prior conceptual knowledge is taken into account during the discovery process in order to find patterns at multiple levels of description granularity. The search through the space of patterns represented as unary conjunctive queries in \mathcal{AL} -LOG and organized according to \mathcal{B} -subsumption is performed by applying an ideal downward refinement operator [31].

2.3. Learning \mathcal{SHIQ} +LOG rules

The ILP framework presented in [27] represents hypotheses as \mathcal{SHIQ} +LOG rules and organizes them according to a generality ordering inspired by generalized subsumption. The resulting hypothesis space can be searched by means of refinement operators either top-down or bottom-up. Analogously to [24], this framework encompasses both scopes of induction but, differently from [24], it assumes the ILP setting of learning from entailment only. Both the coverage relation and the generality relation boil down to query answering in \mathcal{SHIQ} +LOG, thus can be reformulated as satisfiability problems. Compared to [47] and [24], this framework shows an added value which can be summarized as follows. First, it relies on a more expressive DL, *i.e.* \mathcal{SHIQ} . Second, it allows for inducing definitions for new DL concepts, *i.e.* rules with a \mathcal{SHIQ} literal in the head.

⁹The OI bias can be considered as an extension of the UNA from the semantic level to the syntactic one of \mathcal{AL} -LOG. It can be the starting point for the definition of either an equational theory or a quasi-order for constrained DATALOG clauses.

Table 2. Three ILP frameworks suitable for learning onto-relational rules.

	Learning CARIN- \mathcal{ALN} rules [47]	Learning \mathcal{AL} -LOG rules [24]	Learning \mathcal{SHIQ} +LOG rules [27]
prior knowledge	CARIN- \mathcal{ALN} KB	\mathcal{AL} -LOG KB	\mathcal{SHIQ} +LOG KB
ontology language	\mathcal{ALN}	\mathcal{ALC}	\mathcal{SHIQ}
rule language	HCL	DATALOG	DATALOG
hypothesis language	CARIN- \mathcal{ALN} non-recursive rules	\mathcal{AL} -LOG non-recursive rules	\mathcal{SHIQ} +LOG non-recursive rules
target predicate	Horn predicate	DATALOG predicate	\mathcal{SHIQ} /DATALOG predicate
logical setting	interpretations	interpretations/entailment	entailment
scope of induction	prediction	prediction/description	prediction/description
generality order	extension of [2] to CARIN- \mathcal{ALN}	extension of [2] to \mathcal{AL} -LOG	extension of [2] to \mathcal{SHIQ} +LOG
coverage test	CARIN query answering	\mathcal{AL} -LOG query answering	\mathcal{DL} +LOG $^{\neg\vee}$ query answering
ref. operators	n.a.	downward	downward/upward
implementation	unknown	yes, see [33]	no
application	no	yes, see [32]	no

Third, it adopts a more flexible form of integration between the DL and the CL part, *i.e.* the weakly-safe one.

The work reported in [29,25] generalizes the results of [27] to any decidable instantiation of \mathcal{DL} +LOG $^{\neg\vee}$. The following section illustrates how learning \mathcal{DL} +LOG $^{\neg}$ rules can support the evolution of ontologies.

3. Learning Rule-based Definitions of \mathcal{DL} Concepts and Roles with ILP

In KE, Ontology Evolution is the timely adaptation of an ontology to changed business requirements, to trends in ontology instances and patterns of usage of the ontology-based application, as well as the consistent management/propagation of these changes to dependent elements [50]. As opposed to Ontology Modification, Ontology Evolution must preserve the consistency of the ontology. According to [41] one can distinguish between conceptual, specification and representation changes.

In this section we consider the conceptual changes of a \mathcal{DL} ontology due to extensional knowledge (*i.e.*, facts of the instance level of the ontology) previously unknown but classified which may become available. In particular, we consider the task of defining new concepts or roles which provide the intensional counterpart of such extensional knowledge and show how this task can be reformulated as an ORL problem [28]. For example, the new facts $\text{LONER}(\text{Joe})$, $\text{LONER}(\text{Mary})$, and $\text{LONER}(\text{Paul})$ concerning known individuals may raise the need for having a definition of the concept LONER in the ontology. One such definition can be learned from these facts together with prior knowledge about Joe, Mary and Paul, *i.e.* facts concerning them and already available in the ontology. A crucial requirement is that the definition must be expressed as a \mathcal{DL} formula or similar. In the following we provide the means for learning rule-based definitions of \mathcal{DL} concepts/roles in the KR framework of \mathcal{DL} +LOG $^{\neg}$.

3.1. The learning problem

We assume that a \mathcal{DL} ontology $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ is integrated with a DATALOG $^{\neg}$ database Π to form a \mathcal{DL} +LOG $^{\neg}$ KB \mathcal{B} . The problem of inducing rule-based definitions of \mathcal{DL} concepts/roles that do not occur in \mathcal{B} can be formalized as follows.

Definition 1 *Given:*

- a $\mathcal{DL}+\text{LOG}^\neg$ KB \mathcal{B} (background theory)
- a \mathcal{DL} predicate name p (target predicate)
- a set $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$ of \mathcal{DL} assertions that are either true or false for p (examples)
- a set \mathcal{L} of $\mathcal{DL}+\text{LOG}^\neg$ definitions for p (language of hypotheses)

the problem of building a rule-based definition of p is to induce a set $\mathcal{H} \subset \mathcal{L}$ (hypothesis) of $\mathcal{DL}+\text{LOG}^\neg$ rules from \mathcal{E} and \mathcal{B} such that:

Completeness $\forall e \in \mathcal{E}^+ : \mathcal{H}$ covers e w.r.t. \mathcal{B}

Consistency $\forall e \in \mathcal{E}^- : \mathcal{H}$ does not cover e w.r.t. \mathcal{B} .

The background theory \mathcal{B} in Definition 1 can be split into an intensional part \mathcal{K} (i.e., the TBox \mathcal{T} plus $\text{IDB}(\Pi)$) and an extensional part \mathcal{F} (i.e., the ABox \mathcal{A} plus $\text{EDB}(\Pi)$). Also we denote by $P_{\mathcal{C}}(\mathcal{B})$, $P_{\mathcal{R}}(\mathcal{B})$, and $P_{\mathcal{D}}(\mathcal{B})$ the sets of concept, role and DATALOG predicate names occurring in \mathcal{B} , respectively. Note that $p \notin P_{\mathcal{C}}(\mathcal{B}) \cup P_{\mathcal{R}}(\mathcal{B})$.

Example 1 Suppose we have a $\mathcal{DL}+\text{LOG}^\neg$ KB \mathcal{B} (adapted from [45]) built upon the alphabets $P_{\mathcal{C}}(\mathcal{B}) = \{\text{RICH}/1, \text{UNMARRIED}/1\}$, $P_{\mathcal{R}}(\mathcal{B}) = \{\text{WANTS-TO-MARRY}/2, \text{LOVES}/2\}$, and $P_{\mathcal{D}}(\mathcal{B}) = \{\text{famous}/1, \text{scientist}/1, \text{meets}/3\}$ and consisting of the following intensional knowledge \mathcal{K} :

- [A1] $\text{RICH} \sqcap \text{UNMARRIED} \sqsubseteq \exists \text{WANTS-TO-MARRY}^- . \top$
- [A2] $\text{WANTS-TO-MARRY} \sqsubseteq \text{LOVES}$
- [R1] $\text{RICH}(X) \leftarrow \text{famous}(X), \neg \text{scientist}(X)$
- [R2] $\text{happy}(X) \leftarrow \text{famous}(X), \text{WANTS-TO-MARRY}(Y, X)$

and the following extensional knowledge \mathcal{F} :

$\text{UNMARRIED}(\text{Mary})$
 $\text{UNMARRIED}(\text{Joe})$
 $\text{famous}(\text{Mary})$
 $\text{famous}(\text{Paul})$
 $\text{famous}(\text{Joe})$
 $\text{scientist}(\text{Joe})$
 $\text{meets}(\text{Mary}, \text{Paul}, \text{Italy})$
 $\text{meets}(\text{Mary}, \text{Joe}, \text{Germany})$
 $\text{meets}(\text{Joe}, \text{Mary}, \text{Italy})$

that concerns the individuals *Mary*, *Joe*, *Paul*, *Italy*, and *Germany*.

The hypothesis language \mathcal{L} in Definition 1 is given as a set of declarative bias constraints. It allows for the generation of $\mathcal{DL}+\text{LOG}^\neg$ rules starting from three disjoint alphabets $P_{\mathcal{C}}(\mathcal{L}) \subseteq P_{\mathcal{C}}(\mathcal{B})$, $P_{\mathcal{R}}(\mathcal{L}) \subseteq P_{\mathcal{R}}(\mathcal{B})$, and $P_{\mathcal{D}}(\mathcal{L}) \subseteq P_{\mathcal{D}}(\mathcal{B})$. Also we distinguish between $P_{\mathcal{D}}^+(\mathcal{L})$ and $P_{\mathcal{D}}^-(\mathcal{L})$ in order to specify which DATALOG predicates can occur in positive and negative literals, respectively. More precisely, we consider $\mathcal{DL}+\text{LOG}^\neg$ rules of the form

$$p(\vec{X}) \leftarrow r_1(\vec{Y}_1), \dots, r_m(\vec{Y}_m), s_1(\vec{Z}_1), \dots, s_k(\vec{Z}_k), \neg u_1(\vec{W}_1), \dots, \neg u_q(\vec{W}_q) \quad (1)$$

where $m, k, q \geq 0$, $p(\vec{X})$ and each $r_j(\vec{Y}_j)$, $s_l(\vec{Z}_l)$, $u_t(\vec{W}_t)$ is an atom with $r_j \in P_D^+(\mathcal{L})$, $s_l \in P_C(\mathcal{L}) \cup P_R(\mathcal{L})$, and $u_t \in P_D^-(\mathcal{L})$. The admissible rules must be compliant with the following restrictions:

DATALOG-safeness every variable occurring in (1) must appear in at least one of the atoms $r_1(\vec{Y}_1), \dots, r_m(\vec{Y}_m), s_1(\vec{Z}_1), \dots, s_k(\vec{Z}_k)$;

weak \mathcal{DL} -safeness every head variable of (1) must appear in at least one of the atoms $r_1(\vec{Y}_1), \dots, r_m(\vec{Y}_m)$.

which also guarantee that the conditions of linkedness and connectedness, usually assumed in ILP, are satisfied.

Example 2 Suppose that the target predicate is the \mathcal{DL} concept *LONER*. If \mathcal{L}^{LONER} is defined over $P_D^+(\mathcal{L}^{LONER}) \cup P_D^-(\mathcal{L}^{LONER}) \cup P_C(\mathcal{L}^{LONER}) = \{famous/1\} \cup \{happy/1\} \cup \{RICH/1, UNMARRIED/1\}$, then the following $\mathcal{DL}+\text{LOG}^-$ rules

$$\begin{aligned} h_1^{LONER} \quad & LONER(X) \leftarrow famous(X) \\ h_2^{LONER} \quad & LONER(X) \leftarrow famous(X), UNMARRIED(X) \\ h_3^{LONER} \quad & LONER(X) \leftarrow famous(X), \neg happy(X) \end{aligned}$$

belong to \mathcal{L}^{LONER} and represent hypotheses of a definition for *LONER*.

Example 3 Suppose now that the \mathcal{DL} role *LIKES* is the target predicate and the set $P_D^+(\mathcal{L}^{LIKES}) \cup P_C(\mathcal{L}^{LIKES}) \cup P_R(\mathcal{L}^{LIKES}) = \{happy/1, meets/3\} \cup \{RICH/1\} \cup \{LOVES/2, WANTS-TO-MARRY/2\}$ provides the building blocks for the language \mathcal{L}^{LIKES} . The following $\mathcal{DL}+\text{LOG}^-$ rules

$$\begin{aligned} h_1^{LIKES} \quad & LIKES(X, Y) \leftarrow meets(X, Z, Y) \\ h_2^{LIKES} \quad & LIKES(X, Y) \leftarrow meets(X, Z, Y), happy(X) \\ h_3^{LIKES} \quad & LIKES(X, Y) \leftarrow meets(X, Z, Y), RICH(Z) \end{aligned}$$

belonging to \mathcal{L}^{LIKES} can be considered hypotheses of a definition for *LIKES*.

The set \mathcal{E} of examples in Definition 1 contains assertions of the kind $p(\vec{a}_i)$ where p is the target predicate and \vec{a}_i is a tuple of individuals occurring in the ABox \mathcal{A} . Note that, when p is a role name, the tuple \vec{a}_i is a pair $\langle a_i^1, a_i^2 \rangle$ of individuals. We assume $\mathcal{B} \cap \mathcal{E} = \emptyset$. However, a possibly incomplete description of each $e_i \in \mathcal{E}$ is in \mathcal{B} .

Example 4 With reference to Example 2, suppose that the following concept assertions:

$$\begin{aligned} e_1^{LONER} \quad & LONER(Mary) \\ e_2^{LONER} \quad & LONER(Joe) \\ e_3^{LONER} \quad & LONER(Paul) \end{aligned}$$

are examples for the target predicate *LONER*.

Example 5 With reference to Example 3, the following role assertions:

$$\begin{aligned} e_1^{LIKES} \quad & LIKES(Mary, Italy) \\ e_2^{LIKES} \quad & LIKES(Mary, Germany) \\ e_3^{LIKES} \quad & LIKES(Joe, Italy) \end{aligned}$$

can be assumed as examples for the target predicate *LIKES*.

3.2. The ingredients for an ILP solution

In order to solve the learning problem in hand with the ILP methodological approach, the language \mathcal{L} of hypotheses needs to be equipped with (i) a *coverage relation* which defines the mappings from \mathcal{L} to the set \mathcal{E} of examples, and (ii) a *generality order* \succeq such that (\mathcal{L}, \succeq) is a search space.

The definition of a *coverage relation* depends on the representation choice for examples. The normal ILP setting is the most appropriate to the learning problem in hand and can be extended to the $\mathcal{DL}+\text{LOG}^\neg$ framework depicted in Definition 1 as follows.

Definition 2 We say that a rule $h \in \mathcal{L}$ covers (does not cover, resp.) an example $e_i = p(\vec{a}_i) \in \mathcal{E}$ w.r.t. a background theory \mathcal{B} iff $\mathcal{B} \cup h \models p(\vec{a}_i)$ ($\mathcal{B} \cup h \not\models p(\vec{a}_i)$, resp.).

Note that the coverage test can be reduced to query answering w.r.t. a $\mathcal{DL}+\text{LOG}^{\neg\vee}$ KB, which in turn can be reformulated as a satisfiability problem of the KB.

Example 6 With reference to Example 2 and 4, the rule h_1^{LONER} covers the example e_1^{LONER} because all NM-models for $\mathcal{B}' = \mathcal{B} \cup h_1^{\text{LONER}}$ do satisfy *famous(Mary)*. It covers also e_2^{LONER} and e_3^{LONER} for analogous reasons. The rule h_2^{LONER} covers only e_1^{LONER} and e_2^{LONER} whereas h_3^{LONER} covers e_2^{LONER} and e_3^{LONER} .

Example 7 With reference to Example 3 and 5, the rule h_1^{LIKES} covers the example e_1^{LIKES} because all NM-models for $\mathcal{B}' = \mathcal{B} \cup h_1^{\text{LIKES}}$ do satisfy *meets(Mary, Z, Italy)*. It covers also e_2^{LIKES} and e_3^{LIKES} for analogous reasons. The rule h_2^{LIKES} covers only e_1^{LIKES} and e_2^{LIKES} whereas h_3^{LIKES} covers only e_1^{LIKES} and e_3^{LIKES} .

The definition of a *generality order* for hypotheses in \mathcal{L} must consider the peculiarities of the chosen \mathcal{L} . Generalized subsumption, subsequently extended in [49] to deal with NAF literals, is suitable for the problem in hand and can be adapted to the case of $\mathcal{DL}+\text{LOG}^\neg$ rules. In the following we provide a characterization of the resulting generality order, denoted by $\succeq_{\mathcal{K}}$, that relies on the reasoning tasks known for $\mathcal{DL}+\text{LOG}^{\neg\vee}$ and from which a test procedure can be derived.

Definition 3 Let $h_1, h_2 \in \mathcal{L}$ be two $\mathcal{DL}+\text{LOG}^\neg$ rules standardized apart, \mathcal{K} a $\mathcal{DL}+\text{LOG}^\neg$ KB, and σ a Skolem substitution for h_2 with respect to $\{h_1\} \cup \mathcal{K}$. We say that h_1 is more general than h_2 w.r.t. \mathcal{K} , denoted by $h_1 \succeq_{\mathcal{K}} h_2$, iff there exists a ground substitution θ for h_1 such that (i) $\text{head}(h_1)\theta = \text{head}(h_2)\sigma$ and (ii) $\mathcal{K} \cup \text{body}(h_2)\sigma \models \text{body}(h_1)\theta$. We say that h_1 is strictly more general than h_2 w.r.t. \mathcal{K} , denoted by $h_1 \succ_{\mathcal{K}} h_2$, iff $h_1 \succeq_{\mathcal{K}} h_2$ and $h_2 \not\succeq_{\mathcal{K}} h_1$. We say that h_1 is equivalent to h_2 w.r.t. \mathcal{K} , denoted by $h_1 \equiv_{\mathcal{K}} h_2$, iff $h_1 \succeq_{\mathcal{K}} h_2$ and $h_2 \succeq_{\mathcal{K}} h_1$.

Example 8 Let us consider the rules reported in Example 2 up to variable renaming:

$$\begin{array}{ll} h_1^{\text{LONER}} & \text{LONER}(A) \leftarrow \text{famous}(A) \\ h_2^{\text{LONER}} & \text{LONER}(X) \leftarrow \text{famous}(X), \text{UNMARRIED}(X) \end{array}$$

In order to check whether $h_1^{\text{LONER}} \succeq_{\mathcal{K}} h_2^{\text{LONER}}$ holds, let $\sigma = \{X/a\}$ a Skolem substitution for h_2^{LONER} with respect to $\mathcal{K} \cup h_1^{\text{LONER}}$ and $\theta = \{A/a\}$ a ground substitution for h_1^{LONER} . The condition (i) is immediately verified. The condition

$$(ii) \mathcal{K} \cup \{famous(a), UNMARRIED(a)\} \models \{famous(a)\}$$

is a ground query answering problem in $\mathcal{DL}+\text{LOG}^-$. It can be easily proved that all NM-models for $\mathcal{K} \cup \{famous(a), UNMARRIED(a)\}$ satisfy $famous(a)$. Thus, it is the case that $h_1^{LONER} \succeq_{\mathcal{K}} h_2^{LONER}$. The viceversa does not hold. Also, $h_1^{LONER} \succ_{\mathcal{K}} h_3^{LONER}$ and h_3^{LONER} is incomparable with h_2^{LONER} .

Example 9 With reference to Example 3, it can be proved that $h_1^{LIKES} \succ_{\mathcal{K}} h_2^{LIKES}$ and $h_1^{LIKES} \succ_{\mathcal{K}} h_3^{LIKES}$. Conversely, the rules h_2^{LIKES} and h_3^{LIKES} are incomparable. Note that

$$\begin{aligned} h_4^{LIKES} & \quad LIKES(X, Y) \leftarrow meets(X, Z, Y), LOVES(X, Z) \\ h_5^{LIKES} & \quad LIKES(X, Y) \leftarrow meets(X, Z, Y), WANTS-TO-MARRY(X, Z) \end{aligned}$$

also belong to \mathcal{L}^{LIKES} . It can be proved that $h_1^{LIKES} \succ_{\mathcal{K}} h_4^{LIKES}$, $h_1^{LIKES} \succ_{\mathcal{K}} h_5^{LIKES}$, and $h_4^{LIKES} \succ_{\mathcal{K}} h_5^{LIKES}$.

Note that the decidability of $\succ_{\mathcal{K}}$ follows from the decidability of $\mathcal{DL}+\text{LOG}^-$. Also it can be proved that $\succ_{\mathcal{K}}$ is a quasi-order (i.e., it is a reflexive and transitive relation) for $\mathcal{DL}+\text{LOG}^-$ rules, therefore the space $(\mathcal{L}, \succ_{\mathcal{K}})$ can be searched by refinement operators like the following one able to traverse the hypothesis space top down.

Definition 4 Let \mathcal{L} be a $\mathcal{DL}+\text{LOG}^-$ hypothesis language built out of the three finite and disjoint alphabets $P_{\mathcal{C}}(\mathcal{L})$, $P_{\mathcal{R}}(\mathcal{L})$, and $P_{\mathcal{D}}^+(\mathcal{L}) \cup P_{\mathcal{D}}^-(\mathcal{L})$. We define a downward refinement operator ρ^{OR} for $(\mathcal{L}, \succeq_{\mathcal{K}})$ such that, for each $h \in \mathcal{L}$, the set $\rho^{\text{OR}}(h)$ contains all $h' \in \mathcal{L}$ that can be obtained from h by applying one of the following refinement rules:

$$\langle \text{AddDataLit_B}^+ \rangle \text{ body}(h') = \text{body}(h) \cup \{r_{m+1}(\vec{Y}_{m+1})\} \text{ if}$$

1. $r_{m+1} \in P_{\mathcal{D}}^+(\mathcal{L})$
2. $r_{m+1}(\vec{Y}_{m+1}) \notin \text{body}(h)$
3. $\text{var}(\text{head}(h)) \subseteq \text{var}(\text{body}(h'))$

$$\langle \text{AddOntoLit_B} \rangle \text{ body}(h') = \text{body}(h) \cup \{s_{k+1}(\vec{Z}_{k+1})\} \text{ if}$$

1. $s_{k+1} \in P_{\mathcal{C}}(\mathcal{L}) \cup P_{\mathcal{R}}(\mathcal{L})$
2. it does not exist any $s_l(\vec{Z}_l) \in \text{body}(h)$ such that $s_{k+1} \sqsubseteq s_l$
3. $\text{var}(\text{head}(h)) \subseteq \text{var}(\text{body}(h'))$

$$\langle \text{SpecOntoLit_B} \rangle \text{ body}(h') = (\text{body}(h) \setminus \{s_l(\vec{Z}_l)\}) \cup \{s'_l(\vec{Z}_l)\} \text{ if}$$

1. $s'_l \in P_{\mathcal{C}}(\mathcal{L}) \cup P_{\mathcal{R}}(\mathcal{L})$
2. $s'_l \sqsubseteq s_l$

$$\langle \text{AddDataLit_B}^- \rangle \text{ body}(h') = \text{body}(h) \cup \{\neg u_{q+1}(\vec{W}_{q+1})\} \text{ if}$$

1. $u_{q+1} \in P_{\mathcal{D}}^-(\mathcal{L})$
2. $u_{q+1}(\vec{W}_{q+1}) \notin \text{body}(h)$
3. $\vec{W}_{q+1} \subset \text{var}(\text{body}^+(h))$

```

function OR-FOIL( $\mathcal{B}, p, \mathcal{E}^+, \mathcal{E}^-, \mathcal{L}$ ):  $\mathcal{H}$ 
1.  $\mathcal{H} := \emptyset$ 
2. while  $\mathcal{E}^+ \neq \emptyset$  do
3.    $h := \{p(\vec{X}) \leftarrow\}$ ;
4.    $\mathcal{E}_h^- := \mathcal{E}^-$ 
5.   while  $\mathcal{E}_h^- \neq \emptyset$  do
6.      $\mathcal{Q} := \{h' \in \mathcal{L} \mid h' \in \rho^{\text{OR}}(h)\}$ ;
7.      $h := \text{OR-FOIL-CHOOSEBEST}(\mathcal{Q})$ ;
8.      $\mathcal{E}_h^- := \{e \in \mathcal{E}^- \mid \mathcal{B} \cup h \models e\}$ ;
9.   endwhile
10.   $\mathcal{H} := \mathcal{H} \cup \{h\}$ ;
11.   $\mathcal{E}_h^+ := \{e \in \mathcal{E}^+ \mid \mathcal{B} \cup h \models e\}$ ;
12.   $\mathcal{E}^+ := \mathcal{E}^+ \setminus \mathcal{E}_h^+$ 
13. endwhile
14. return  $\mathcal{H}$ 

```

Figure 1. OR-FOIL: A FOIL-like algorithm for learning onto-relational rules

All the rules of ρ^{OR} are correct, *i.e.* the h 's obtained by applying any of the rules of ρ^{OR} to $h \in \mathcal{L}$ are such that $h \succ_{\mathcal{K}} h'$. This can be proved intuitively by observing that they act only on $\text{body}(h)$. Thus condition (i) of Definition 3 is satisfied. Furthermore, it is straightforward to notice that the application of any of the rules of ρ^{OR} to h reduces the number of models of h . In particular, as for $\langle \text{SpecOntoLit}_B \rangle$, this intuition follows from the semantics of DLs. So condition (ii) also is fulfilled.

Example 10 With reference to Example 2, applying $\langle \text{AddDataLit}_B^+ \rangle$ to

$h_0^{\text{LONER}} \quad \text{LONER}(X) \leftarrow$

produces h_1^{LONER} which can be further specialized by means of $\langle \text{AddOntoLit}_B \rangle$ and $\langle \text{AddDataLit}_B^- \rangle$. Note that no other refinement rule can be applied to h_1^{LONER} and that h_2^{LONER} and h_3^{LONER} are among the refinements of h_1^{LONER} .

Example 11 With reference to Example 3, applying $\langle \text{AddDataLit}_B^+ \rangle$ to

$h_0^{\text{LIKES}} \quad \text{LIKES}(X, Y) \leftarrow$

produces h_1^{LIKES} which can be further specialized into h_2^{LIKES} , h_3^{LIKES} , h_4^{LIKES} and h_5^{LIKES} by means of $\langle \text{AddDataLit}_B \rangle$ and $\langle \text{AddOntoLit}_B \rangle$. Note that no other refinement rule can be applied to h_1^{LIKES} and that h_5^{LIKES} can be also obtained as refinement from h_4^{LIKES} via $\langle \text{SpecOntoLit}_B \rangle$.

3.3. An ILP algorithm

The ingredients identified in the previous section are the starting point for the definition of ILP algorithms. Figure 1 reports the main procedure of a FOIL-like algorithm, named OR-FOIL, for learning onto-relational rules. In OR-FOIL, analogously to FOIL¹⁰, the

¹⁰FOIL is a popular ILP algorithm for learning sets of rules to be used as a classifier [42].

outer loop (steps 2-12) corresponds to a variant of the sequential covering algorithm, *i.e.*, it learns new rules one at a time, removing the positive examples covered by the latest rule before attempting to learn the next rule (steps 11-12). The hypothesis space search performed by OR-FOIL is best understood by viewing it hierarchically. Each iteration through the outer loop (steps 2-13) adds a new rule to its disjunctive hypothesis \mathcal{H} . The effect of each new rule is to generate the current disjunctive hypothesis (*i.e.*, to increase the number of instances it classifies as positive), by adding a new disjunct. Viewed at this level, the search is a bottom-up search through the space of hypotheses, beginning with the most specific empty disjunction (step 1) and terminating when the hypothesis is sufficiently general to cover all positive training examples (step 13). The inner loop (steps 5-9) performs a more fine-grained search to determine the exact definition of each new rule. This loop searches a second hypothesis space, consisting of conjunctions of literals, to find a conjunction that will form the body of the new rule. Within this space, it conducts a top-down, hill-climbing search, beginning with the most general preconditions possible (step 3), then refining the rule (step 6) until it avoids all negative examples. To select the most promising specialization from the candidates generated at each iteration, OR-FOIL-CHOOSEBEST (called at step 7) considers the performance of each candidate over \mathcal{E} and chooses the one which maximizes the *information gain*. This measure is computed according to the following formula

$$\text{GAIN}(h', h) = p * (\log_2(cf(h')) - \log_2(cf(h))) , \quad (2)$$

where p is the number of distinct variable bindings with which positive examples covered by the rule h are still covered by h' and $cf()$ is the confidence degree. Thus, the gain is positive iff h' is more informative in the sense of Shannon's information theory (*i.e.* iff the confidence degree increases). If there are some literals to add which increase the confidence degree, the information gain tends to favor the literals that offer the best compromise between the confidence degree and the number of examples covered.

One may think to use the confidence degree defined for \mathcal{DL} -FOIL (see Chapter ?? for more details) which takes OWA into account. Indeed, many individuals may be available which can not be classified as instances of the target concept nor of its negation. This requires a different setting able to deal with unlabeled individuals.

Example 12 *With reference to Example 10 and Example 6, we suppose that*

$$\begin{aligned} \mathcal{E}^+ &= \{e_1^{LONER}, e_2^{LONER}\} \\ \mathcal{E}^- &= \{e_3^{LONER}\} \end{aligned}$$

The outer loop of OR-FOIL starts from h_0^{LONER} which is further refined through the iterations of the inner loop, more precisely it is first specialized into h_1^{LONER} which in turn, since it covers negative examples, is then specialized into h_2^{LONER} and h_3^{LONER} out of which the rule h_3^{LONER} is added to \mathcal{H}^{LONER} the hypothesis because it does not cover negative examples. At this point the algorithm stops because \mathcal{H}^{LONER} covers both positive examples.

Example 13 *Following Example 11 and Example 7, we assume that $\mathcal{E}^+ = \{e_1^{LIKES}, e_3^{LIKES}\}$ and $\mathcal{E}^- = \{e_2^{LIKES}\}$. At the end of the first iteration, h_3^{LIKES} is included into \mathcal{H}^{LIKES} since it does not cover negative examples but only one positive example.*

4. Final Remarks and Directions of Research

Building rules within ontologies poses several challenges not only to KR researchers investigating suitable hybrid DL-CL formalisms but also to the ML community which has been historically interested in application areas where the knowledge acquisition bottleneck is particularly severe. In particular, ORL may open up new opportunities for KE because it will make systems available to support the knowledge engineer in her most demanding task, *i.e.* defining rules that extend or complement an ontology. Thus, ORL may produce time and cost savings in KE. In this chapter, we have revised the ML literature addressing the problem of learning onto-relational rules. Very few ILP works have been found that propose a solution to this problem [47,24,27]. They adopt CARIN- \mathcal{ALN} , \mathcal{AL} -LOG and \mathcal{SHIQ} +LOG as KR framework, respectively. Note that matching Table 2 against Table 1 one may figure out what is the state-of-the-art and what are the directions of research on onto-relational rules from the ML viewpoint. Also he/she can get suggestions on what is the most appropriate among these ILP frameworks to be implemented for a certain intended application. The specific solution illustrated in Section 3 takes advantage from an augmented expressive power thanks to the chosen \mathcal{DL} +LOG[∨] instantiation [25]. It supports the evolution of ontologies with the creation of a concept/role, change operations which both boil down to the addition of new rules to the input KB.

From the comparative analysis of the ILP frameworks reviewed in Section 2, a common feature emerges: All proposals resort to Buntine's generalized subsumption and extend it in a non-trivial way. This choice is due to the fact that, among the semantic generality orders in ILP, generalized subsumption applies only to definite clauses, therefore suits well the hypothesis language in all three frameworks. Following these guidelines, new ILP frameworks can be designed to deal with more or differently expressive hybrid DL-CL languages according to the DL chosen (*e.g.*, learning CARIN- \mathcal{ALCN} rules), or the clausal language chosen (*e.g.*, learning recursive CARIN rules), or the integration scheme (*e.g.*, learning CARIN rules with \mathcal{DL} -literals in the head). An important requirement will be the definition of a *semantic* generality relation for hypotheses to take into account the background knowledge. Of course, generalized subsumption may turn out to be not suitable for all cases, *e.g.* for the case of learning \mathcal{DL} +LOG[∨] rules [25]. Also it would be interesting to investigate how the nature of rules (*i.e.*, the intended context of usage) may impact the learning process as for the scope of induction and other variables in the learning problem statement. For example, the problem of learning \mathcal{AL} -LOG rules for classification purposes differ greatly from the apparently similar learning problem faced in [32]. Finally, it is worthy to consider hybrid KR formalisms with loose and full integration scheme.

Besides theoretical issues, most future work will have to be devoted to implementation and application. When moving to practice, issues like efficiency and scalability become of paramount importance. These concerns may drive the attention of ILP research towards less expressive hybrid KR frameworks in order to gain in tractability, *e.g.* instantiations of \mathcal{DL} +LOG[∨] with DL-Lite [4]. Applications can come out of some of the many use cases for Semantic Web rules specified by the RIF W3C Working Group.

References

- [1] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.
- [2] W. Buntine. Generalized subsumption and its application to induction and redundancy. *Artificial Intelligence*, 36(2):149–176, 1988.
- [3] J. Cabral, R.C. Kahlert, C. Matuszek, M.J. Witbrock, and B. Summers. Converting semantic meta-knowledge into inductive bias. In S. Kramer and B. Pfahringer, editors, *Inductive Logic Programming*, volume 3625 of *Lecture Notes in Computer Science*, pages 38–50. Springer, 2005.
- [4] D. Calvanese, M. Lenzerini, R. Rosati, and G. Vetere. DL-Lite: Practical Reasoning for Rich DLs. In V. Haarslev and R. Möller, editors, *Proc. of the 2004 Int. Workshop on Description Logics*, volume 104 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.
- [5] S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1(1):146–166, 1989.
- [6] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer, 1990.
- [7] L. De Raedt. *Logical and Relational Learning*. Springer, 2008.
- [8] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. \mathcal{AL} -log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.
- [9] W. Drabent, T. Eiter, G. Ianni, T. Krennwallner, T. Lukasiewicz, and J. Maluszynski. Hybrid Reasoning with Rules and Ontologies. In F. Bry and J. Maluszynski, editors, *Semantic Techniques for the Web, The REWERSE Perspective*, volume 5500 of *LNCSE*, pages 1–49. Springer, 2009.
- [10] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive DATALOG. *ACM Transactions on Database Systems*, 22(3):364–418, 1997.
- [11] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the semantic web. *Artificial Intelligence*, 172(12–13):1495–1539, 2008.
- [12] T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In *IJCAI*, pages 90–96, 2005.
- [13] A.M. Frisch. The substitutional framework for sorted deduction: Fundamental results on hybrid reasoning. *Artificial Intelligence*, 49:161–198, 1991.
- [14] A.M. Frisch. Sorted downward refinement: Building background knowledge into a refinement operator for inductive logic programming. In S. Džeroski and P. Flach, editors, *Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 104–115. Springer, 1999.
- [15] A.M. Frisch and A.G. Cohn. Thoughts and afterthoughts on the 1988 workshop on principles of hybrid reasoning. *AI Magazine*, 11(5):84–87, 1991.
- [16] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.
- [17] B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic *SHIQ*. *Journal of Artificial Intelligence Research*, 31:151–198, 2008.
- [18] F. Goasdoué, V. Lattès, and M.-C. Rousset. The Use of CARIN Language and Algorithms for Information Integration: The PICSEL System. *Int. J. Cooperative Information Systems*, 9(4):383–401, 2000.
- [19] I. Horrocks, J. Angele, S. Decker, M. Kifer, B. Grosz, and G. Wagner. Where are the rules? *IEEE Intelligent Systems*, 18:76–83, 2003.
- [20] J. Józefowska, A. Lawrynowicz, and T. Lukaszewski. The role of semantics in mining frequent patterns from knowledge bases in description logics with rules. *Theory and Practice of Logic Programming*, 10(3):251–289, 2010.
- [21] J.-U. Kietz. Learnability of description logic programs. In S. Matwin and C. Sammut, editors, *Inductive Logic Programming*, volume 2583 of *LNAI*, pages 117–132. Springer, 2003.
- [22] D.B. Lenat, R.V. Guha, K. Pittman, D. Pratt, and M. Shepherd. Cyc: Toward programs with common sense. *Commun. ACM*, 33(8):30–49, 1990.
- [23] A.Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104:165–209, 1998.
- [24] F.A. Lisi. Building Rules on Top of Ontologies for the Semantic Web with Inductive Logic Programming. *Theory and Practice of Logic Programming*, 8(03):271–300, 2008.
- [25] F.A. Lisi. Inductive Logic Programming in Databases: From Datalog to \mathcal{DL} +log. *Theory and Practice of Logic Programming*, 10(3):331–359, 2010.
- [26] F.A. Lisi and F. Esposito. Efficient Evaluation of Candidate Hypotheses in \mathcal{AL} -log. In R. Camacho, R. King, and A. Srinivasan, editors, *Inductive Logic Programming*, volume 3194 of *Lecture Notes in*

- Artificial Intelligence*, pages 216–233. Springer, 2004.
- [27] F.A. Lisi and F. Esposito. Foundations of Onto-Relational Learning. In F. Železný and N. Lavrač, editors, *Inductive Logic Programming*, volume 5194 of *Lecture Notes in Artificial Intelligence*, pages 158–175. Springer, 2008.
 - [28] F.A. Lisi and F. Esposito. Learning *SHIQ*+log Rules for Ontology Evolution. In A. Gangemi, J. Keizer, V. Presutti, and H. Stoermer, editors, *Semantic Web Applications and Perspectives (SWAP2008)*, volume 426 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
 - [29] F.A. Lisi and F. Esposito. Nonmonotonic Onto-Relational Learning. In L. De Raedt, editor, *Inductive Logic Programming*, volume 5989 of *Lecture Notes in Computer Science*, pages 88–95, 2010.
 - [30] F.A. Lisi and D. Malerba. Bridging the Gap between Horn Clausal Logic and Description Logics in Inductive Learning. In A. Cappelli and F. Turini, editors, *AI*IA 2003: Advances in Artificial Intelligence*, volume 2829 of *Lecture Notes in Artificial Intelligence*, pages 49–60. Springer, 2003.
 - [31] F.A. Lisi and D. Malerba. Ideal Refinement of Descriptions in *AL*-log. In T. Horvath and A. Yamamoto, editors, *Inductive Logic Programming*, volume 2835 of *Lecture Notes in Artificial Intelligence*, pages 215–232. Springer, 2003.
 - [32] F.A. Lisi and D. Malerba. Inducing Multi-Level Association Rules from Multiple Relations. *Machine Learning*, 55:175–210, 2004.
 - [33] F.A. Lisi. *AL*-QUIN: An Onto-Relational Learning System for Semantic Web Mining. *Int. J. Semantic Web and Information Systems*, 7(3):1–22, 2011.
 - [34] J.W. Lloyd. *Foundations of Logic Programming*. Springer, 2nd edition, 1987.
 - [35] V.W. Marek and M. Truszczyński. Stable models and an alternative logic programming paradigm. In K.R. Apt, V.W. Marek, M. Truszczyński, and D.S. Warren, editors, *The Logic Programming Paradigm: a 25-Year Perspective*, pages 169–181. Springer, 1999.
 - [36] T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.
 - [37] B. Motik and R. Rosati. Reconciling description logics and rules. *J. ACM*, 57(5), 2010.
 - [38] B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with Rules. *Journal on Web Semantics*, 3(1):41–60, 2005.
 - [39] S. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–317, 1991.
 - [40] S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer, 1997.
 - [41] N. Fridman Noy and M.C.A. Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, 6(4):428–440, 2004.
 - [42] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.
 - [43] R. Rosati. On the decidability and complexity of integrating ontologies and rules. *Journal of Web Semantics*, 3(1):61–73, 2005.
 - [44] R. Rosati. Semantic and computational advantages of the safe integration of ontologies and rules. In F. Fages and S. Soliman, editors, *Principles and Practice of Semantic Web Reasoning*, volume 3703 of *Lecture Notes in Computer Science*, pages 50–64. Springer, 2005.
 - [45] R. Rosati. *DL*+log: Tight Integration of Description Logics and Disjunctive Datalog. In P. Doherty, J. Mylopoulos, and C.A. Welty, editors, *Proc. of Tenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 68–78. AAAI Press, 2006.
 - [46] R. Rosati. On Combining Description Logic Ontologies and Nonrecursive Datalog Rules. In D. Calvanese and G. Lausen, editors, *Web Reasoning and Rule Systems*, volume 5341 of *Lecture Notes in Computer Science*, pages 13–27. Springer, 2008.
 - [47] C. Rouveirol and V. Ventos. Towards Learning in CARIN-*ALN*. In J. Cussens and A. Frisch, editors, *Inductive Logic Programming*, volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 191–208. Springer, 2000.
 - [48] E. Ruckhaus, V. Kolovski, B. Parsia, and B. Cuenca Grau. Integrating Datalog with OWL: Exploring the *AL*-log Approach. In S. Etalle and M. Truszczyński, editors, *Logic Programming*, volume 4079 of *Lecture Notes in Computer Science*, pages 455–456. Springer, 2006.
 - [49] C. Sakama. Nonmonotonic inductive logic programming. In T. Eiter, W. Faber, and M. Truszczyński, editors, *Logic Programming and Nonmonotonic Reasoning*, volume 2173 of *Lecture Notes in Computer Science*, pages 62–80. Springer, 2001.
 - [50] L. Stojanovic, A. Maedche, B. Motik, and N. Stojanovic. User-driven ontology evolution management. In A. Gómez-Pérez and V.R. Benjamins, editors, *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, volume 2473 of *LNCS*, pages 285–300. Springer, 2002.